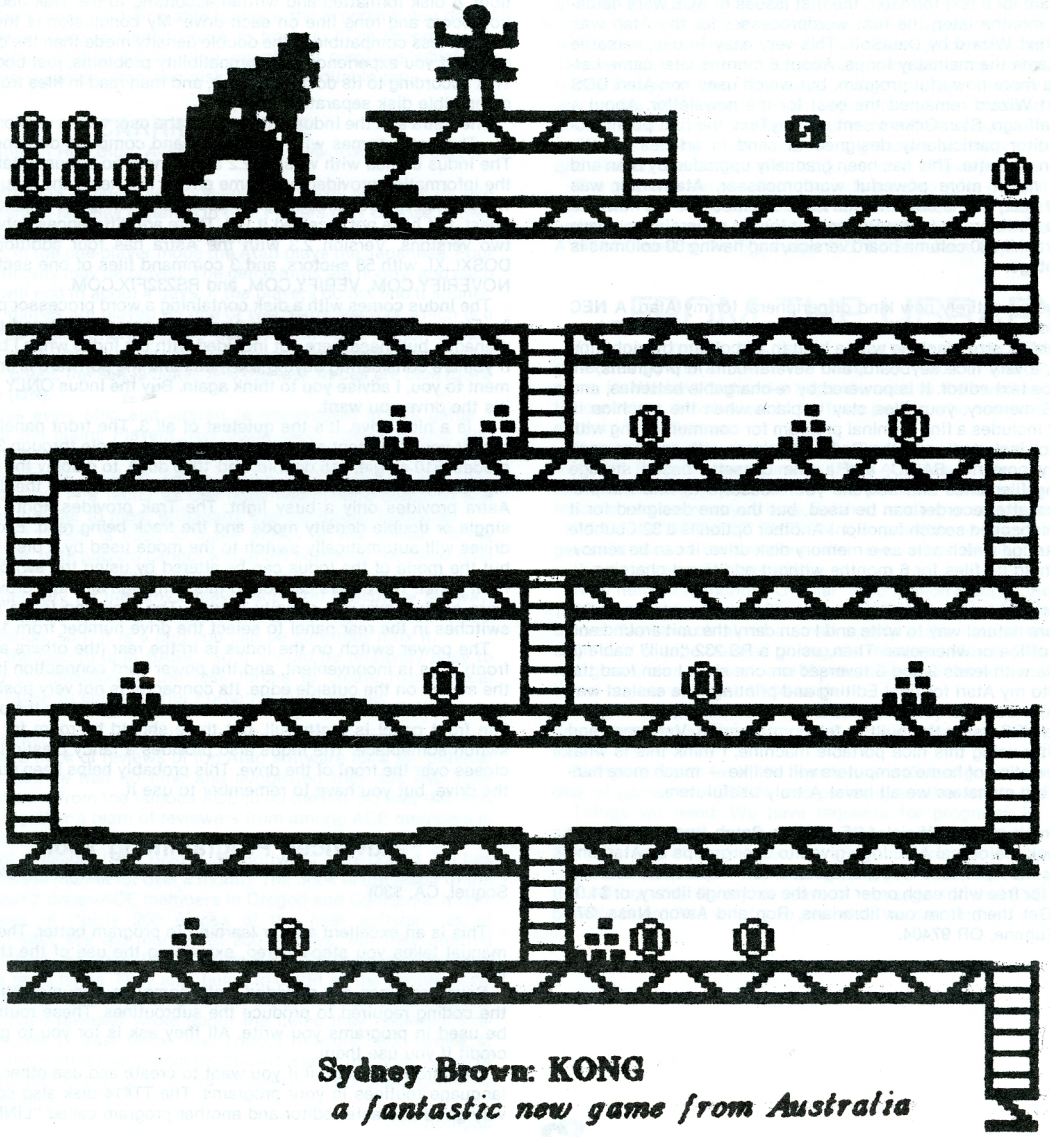
ATARI COMPUTER ENTHUSIASTS

3662 Vine Maple Dr. Eugene OR 97405

MARCH, 1984
Mike Dunn & Jim Bumpas, Editors



Sydney Brown: KONG
a fantastic new game from Australia

News and Reviews

by Mike Dunn, Co-Editor

This month we continue having some very nice games. Sydney Brown from Australia has sent a number of very fine games from Australia for us. The first is **KONG** which is of commercial quality, as are many of his games. If you have a compilier, the game might be as good as any on the market. Another fine game from Stan Ockers, as usual very nicely done, in ACTION! This game is also a very nice climbing type game. Also in ACTION! is a Demo program from the author of ACTION!, Clint Parker. More programs in PILOT, and some machine language utilities.

I notice the last issue of **Softside** not only doesn't have anything at all for Atari, there are no listings at all — no reason given either. Since most of us subscribe to Softside for the programs, and there are no programs, I wonder what it all means. I recieved a very beautiful book from Reston/ Ashton-Tate, **Through the Micromaze, a Visual Guide from Ashton-Tate**, by Wayne Creekmore ($10). An outstanding work of art, it is a beginners guide to computers and their applications. Designed for the business computer user, I could recommend it for any one.

For the first time since I have had an Atari, I am writing an article on a non-Atari computer. When we began the ACE Newsletter about 4 years ago, I had to write a simple word-processor since none for the Atari had been released. Using the DOS as an Editor and a simple BASIC program for a text formater, the first issues of ACE were hatched. About 9 months later, the first wordprocessor for the Atari was developed, **Text Wizard** by DataSoft. This very easy to use, versatile program became the mainstay for us. About 6 months later came **Letter Perfect**, a more powerful program, but which uses non-Atari DOS files, so Text Wizard remained the best for the newsletter. About a year and a half ago, Stan Ockers sent us **TinyText**, the first public domain text editor particularily designed to send in articles for the newsletter on cassette. This has been gradually upgraded by Stan and others to a much more powerful wordprocessor. **AtariWriter** was released last year, and is now the one I use the most for the newsletter, although I still prefer Letter Perfect for longer papers. I now have the Austin Franklin 80 column board version, and having 80 columns is a nice advantage.

Now I have an entirely new kind of peripheral for my Atari. A **NEC 8201A** Lap computer, very similiar the Radio Shack Model 100. This is a small notebook size machine with a built in 40 column by eight line LCD screen, a very nice keyboard, and several built in programs, including a nice text editor. It is powered by re-chargable batteries, and using CMOS memory, your files stay in place when the machine is turned off. It includes a fine terminal program for communicating with a modem (not included as in the Radio Shack) or with another computer, a very powerful BASIC, and a fine cassette based storage system using filenames and searchs your cassette to find the program. Any casette recorder can be used, but the one designed for it even has a highspeed search function! Another option is a 32K bubble memory cartridge which acts as a memory disk drive. It can be removed and will hold its files for 6 months without additional charging.

What does all this have to do with my Atari? I find the screen on the desk is a more natural way to write and I can carry the unit around and write in my office or whereever. Then, using a RS-232 "null" cable (a modem cable with leads 2 and 3 reversed on one end), I can load the files right into my Atari for final Editing and printing. The easiest way to do this is to use Atari Writer, and "Load" from "R1:", sending the file from the NEC using the built in telecom program. Very easy and very fast. After using this nice portable machine, I think this is what the next generation of home computers will be like — much more handy than the big monsters we all have! A truly useful item.

You can now get an official **ACE Iron-On Patch** from us. George Suetsugu took the offical ACE logo given to user groups by Atari and our cover for the April 1983 issue, and made iron-on patches. One will be included for free with each order from the exchange library, or $1.00 for extras. Get them from our librarians, Ron and Aaron Ness, 374 Blackfoot, Eugene, OR 97404.

BUMPAS REVIEWS

More on the new drives:

With the help of the folks at Computer Palace and my friend Nick Chrones, I've had the opportunity to play around with the 3 new drives: the **ASTRA 1620**, the **TRAK AT-D2**, and the **INDUS GT**. I'm personally using the Indus, but my opinion of the other two drives has improved over the past couple of weeks.

I'm not going to discuss the compatibility problems caused by commercial software which use illegal entry points to the operating system. I think the Indus is a little more compatible than the other two, but I admit this opinion is 2d hand. I am going to discuss compatibility problems between the drives in double density mode. In single density mode each of the drives can read and boot a disk formatted and written with any of the other drives.

In double density mode, I have found some problem booting disks on one drive which were formatted and written on another drive. I did not discover any problem in this area on the Astra. And disks formatted and written on the Astra seem to boot fine on either the Trak or the Indus. And disks formatted and written on the Trak seem to boot perfectly well on the Indus. However, several disks formatted and written on the Indus failed to boot on the Trak. We tried several operating systems, including DOS 2.0, DOS XL versions 2.2 and 2.3, and MYDOS version 3.5.

We found we could boot a DOS on the Trak and then read files on another disk. These files run just fine. Next, we formatted a disk and wrote the TADS modified DOS 2.0 (explained in the Trak documentation). A disk formatted and written according to the Trak documentation boots and runs fine on each drive. My conclusion is the Trak is slightly less compatible in the double density mode than the other two drives. If you experience any compatibility problems, just boot up the Trak according to its documentation, and then read in files from the incompatible disk separately.

The Astra and the Indus both provide the user with a version of DOS XL. The Astra comes with version 2.3 and complete documentation. The Indus comes with version 2.2 and almost no documentation, and the information provided has some errors. I've heard the Indus people will provide full documentation to users requesting it. I've made my request, but have not received it yet. There are differences between the two versions. Version 2.3 with the Astra has four additional files: DOSXL.XL with 58 sectors, and 3 command files of one sector each: NOVERIFY.COM, VERIFY.COM, and RS232FIX.COM.

The Indus comes with a disk containing a word processor produced by Elcomp. They also promise to send a spreadsheet and database manager, but these were not included with the Indus when I bought it. If you are considering buying the Indus and the software is an inducement to you, I advise you to think again. Buy the Indus ONLY because it's the drive you want.

It is a nice drive. It's the quietest of all 3. The front panel buttons permit you to protect an unprotected disk, to cycle through 3 density modes (810 single, 815 double, and 1050 dual), to display the track being read, and the error number of any error detected by the drive. The Astra provides only a busy light. The Trak provides lights to show single or double density mode and the track being read. Each of the drives will automatically switch to the mode used by a disk inserted, but the mode of the Indus can be altered by using the switch on the front panel. The others cannot do this, although all three provide software control over the density modes. The Trak and Indus have dip switches in the rear panel to select the drive number from 1-4.

The power switch on the Indus is in the rear (the others are in the front). This is inconvenient, and the power cord connection is next to the switch on the outside edge. Its connector is not very positive, so I find myself disconnecting the power when trying to turn the power on. The front panel is pretty full, but there should be room for a power switch someplace. The Indus also provides a fancy plastic lid which closes over the front of the drive. This probably helps keep dust out of the drive, but you have to remember to use it.

Advanced Programming Tools

(Tricky Tutorial no. 14, by Educational Software, 4565 Cherryvale Ave., Soquel, CA, $30)

This is an excellent aid for learning to program better. The 51-page manual takes you step-by-step, explaining the use of the USR function.

Seven routines are provided, with complete documentation of all the coding required to produce the subroutines. These routines may be used in programs you write. All they ask is for you to give them credit if you use them.

The tutorial is helpful if you want to create and use other machine language routines in your programs. The TT#14 disk also contains a very good character editor and another program called "LINKBAS".

This program alone makes the disk a worthwhile addition to anyone's library. LINKBAS takes binary data from a compiled assembly language routine, moves it to a string variable to be used as a USR call in a BASIC program, and even generates the actual BASIC code automatically! All the user has to do is ENTER the generated code into the program.

These tutorials are getting better and better. ESI deserves our appreciation for these excellent programs.

The Programmer's Toolkit

(Sure Soft, 8177 S. Harvard, suite 428, Tulsa, OK 74137, $30)

This program is a little misleading. The documentation suggests the user consider this disk as something akin to a toolbox used by a carpenter to build a house. However, this toolbox must be left inside each house the carpenter builds.

The disk contains over 30 machine language routines and about 40 demonstration programs for using the routines. The advertizing and introduction to the documentation lead one to believe one may use these machine language routines in one's programs. Well, you can, if you boot up the Toolkit disk before you load and run any BASIC program using the supplied routines.

So, actually the programmer is not putting these routines into any program one might write. Users will not learn much about programming by using this disk unless the object code can be disassembled to study the machine language routines. But if the user has the skill to do this, he probably already has his own personal "tool kit" of most of these routines. Some of the more interesting routines include a string editor, right and left justification, and a routine which searches a string for the occurence of a sequence of 1 to 256 characters. This is a good program to excite a user about the possibilities of machine language routines in BASIC programs.

If you don't mind not being able to share the programs you design with others and are content to have to boot up "The Programmer's Toolkit" before loading and running any of your programs which use the routines supplied, then this disk does a creditable job.

CARRIER FORCE

(SSI, 883 Stierlin Rd., Bldg. A-200, Mountain View, CA 94043, $60)

Carrier Force is a fun-to-play hide-and-seek naval wargame for one or two players. It is really 4 games in one. Players may choose to play one of four campaigns: Coral Sea; Midway; Eastern Solomons; and Santa Cruz. In the one-player mode the Atari plays the Japanese. Four levels of difficulty may be played. The game-turns are hourly, and each campaign will end in about 4 days, if the players do not end it sooner.

The four games are played on one of two maps, each of which is a grid of hexes approximately 30x30 covering almost 4 million square kilometers each. The maps scroll across approximately 8 screens. The islands are green, the sea is black, and the task forces are blue (USN) and yellow (IJN).

SSI claims every ship and aircraft is represented in this game. I don't doubt the claim. Players may move each individual ship and plane. Aircraft may be armed with bombs for attacking either ships or land bases. Planes must be readied on the runway before they may take off. Weather is a factor as one must watch wind direction and cloud cover in order to most efficiently use the forces available.

A large red cursor is used to make moves on the screen. In the 2-player mode, both players cannot watch the screen at the same time except during the joint combat phase at the end of each turn. I've been unable to get very far into any 2-player scenario except for Midway, because it crashes on me. But the solitaire games work fine. SSI has a good reputation, so if it is a bug you can be sure they'll fix it.

Most of the coding is in BASIC, so it should be relatively easy to modify the composition of the task forces to create hypothetical scenarios. I plan to enjoy Carrier Force for quite awhile.

Last summer, **Consumer Guide** magazine called Mike Dunn to ask him to write a book of reviews of top Atari software for a fall publication. Mike said he had no time and suggested they ask me. They wanted someone from the famous ACE to do the job. So they did. And I did. I put together a team of reviewers from among ACE members in California and Oregon and we finished the work in less than 3 months. The book of reviews of the Best Atari Software (wire-bound, $5) has now been on the market for over a month. The book is the result of the work of about 2 dozen ACE members in Oregon and California. It contains reviews of nearly 200 pieces of the best software as of September, 1983. We did more than 200 reviews, but they omitted about 30 of them, including Data Perfect, Armudic, Synassembler, and some other good items. They also failed to credit all the people working on the project. Even my name appears only in small print on an inside page. Each reviewer wrote about the items they reviewed. I rewrote each one at least twice. And editors re-wrote each review at least twice more after I sent them off. Nevertheless, the book is one of the most helpful of its type on the market — especially given its reasonable price.

— Jim Bumpas

THE RETURN OF HERACLES

Quality Software $33

The Return of Heracles is a graphics adventure exploring Greek mythology. One to four players may become an ancient Greek hero, heroine, or even a winged horse. You move your character around in ancient Greece, gathering gold, slaying opponents, sharpening skills, and trying to complete the twelve tasks set by Zeus, head god of Olympus. Several heroes may work together, and a single player may control several heroes at once. There are nineteen heroes at the beginning, but others may be found along the way. But once a hero is dead, he can't be resurrected.

The Oracle of Zeus is where characters discover the twelve tasks of Zeus. Players must travel to the Oracle and approach the sacred tree of Zeus. Then he reveals a task. Tasks do not have to be done in the order assigned by Zeus. The Oracle of Delphi can help the players. Players must first travel there, and for a suitable offering of drackmae (gold), she will give a clue to help. Sometimes the clue isn't for the task Zeus has currently assigned. If players can't handle the current task, they can go back to the Oracle of Zeus and get a new one.

An interesting feature is that, after start, the game is almost totally controlled by joystick. The exception is the SYSTEM RESET button. Pressing this will start the game from the very beginning (if THE RETURN OF HERACLES disk is in drive #1). You can use the joystick to move your character, stop, rest, defend or attack. Using the KNOW THYSELF option, a player can examine his traits, his weapons, and his armor. Other commands include add hero, desert, change message or monster speed, save game, and drop armor or gold.

The combat system is quite good. By using probabilities based on dexterity, weapon skill, size, and speed, a well-rounded combat system is created. A touch of luck is included as characters always have a small chance to hit or miss. Also, weapons can break at the wrong time, or a lucky hit can do more damage than normally possible. The two types of combat are normal (using a sword), and hand-to-hand (using a dagger). You must be adjacent to a player in order to attack it, and in the same space as an opponent for hand-to-hand. You must always attack a hand-to-hand opponent before an adjacent one.

The instruction book includes a glossary of important characters and places, and a chart with ability scores of everybody in the game.

The graphics are excellent. Sounds are also quite good. It is obvious the writer did some research for all the names and places. It is very well done. This is for the true adventurer, but at the intermediate level. A bit of Greek mythology helps too. An excellent buy.

— Aaron Ness

FROM THE LIBRARY

This past month has, again, been a busy one. The inquiries generated by the article in the December issue of Antic continue to come in. We have noticed a large increase in the number of requests for cassettes. Most seem to be from people who received their computers for Christmas.

For those of you who are new to the computer, we will pass along this hint from the Jacksonville (Fla.) ACE Newsletter. Don't take a chance on ruining your tape or the pressure roller in your cassette recorder. As soon as you have finished loading or saving that program - HIT THE STOP/EJECT BUTTON. It is possible to "dent" the tape or get a flat spot on the pressure roller if you leave the play button down without the recorder operating.

New items this month: "Official" ACE iron-on patches. See the April 1983 newsletter cover. These are about 3 inches square and come in 4 colors. We have obtained a small supply and will send **one free** with each disk or tape ordered during the month of March. Please specify red, green, blue, or black and **if possible** we will send that color. Additional copies will cost $1.00 each.

Thanks to the efforts of Jim Carr and Deloy Graham, we now have documentation available for the Source Codes Disk and Utility Disk #2.

Things to watch for in the near future: a disk of goodies in FORTH, a disk of games in ACTION, and perhaps, another utility disk.

Things we need: We have requests for programs in chemistry, physics, and math programs for geometry and above. If you know of any programs of these types that are public domain, let's get together.

Let us know what kind of programs you need. Just drop a note to the editors, the library or leave a message for the SYSOP on the bulletin board. We will try to see what we can find.

— Ron Ness

ERACE

The Incredible Laboratory

Strategies in Problem Solving

The Incredible Laboratory (Sunburst Communications, $49) gives you the chance to create your own monsters (some quite silly) by combining chemicals from lists provided. Each chemical will alter the appearance of a body part, such as head, eyes, arms, etc. There are six different possibilities for each body part so there are many possible combinations to try and produce.

Even though making a particular monster is very challenging, there is much more to be learned from this program. The Incredible Laboratory was designed to help students learn about the trial and error process and to help them develop good note taking skills.

In the lower level of play, a given chemical will produce a constant effect, and it is up to the student to determine what the effect is. By selecting chemicals, or by leaving them out of the "foumula", the effect of the different chemicals can be learned. Good note taking will be a big help in keeping things in order.

When you think you know how to control your experiments and make a particular monster, the challenge mode of the game can be used to test your skills.

The graphic drawings of the monsters are creative without being gruesome. The laboratory and the monsters are very well done and add to the quality of the program.

There are five different levels of play from novice to scientist. In the upper levels more chemicals are used. They may be combined to produce new effects, and each time the program is run their effects change. The challenge round can accommodate two players, and each of the three main levels has a challenge round.

The ground rules given as the instructions at the beginning of the game are not very complete and can lead to some some confusion of the objectives of the game portion of the program. But the teacher's guide has all the information needed to run the program and get the most out of it. There are several aids and lessons included that will help introduce the student to the processes of trial and error and organized note taking. Although the program is intended to be used in the classroom, with the help of a parent it can be just as effective in the home. I strongly suggest the manual be read and the supporting material used as needed to get the most out of this program.

As with the other programs I have seen from Sunburst Communications, the documentation of the program is outstanding. If the program is used as suggested in the teacher's guide a great deal can be learned while making monsters.

Some criticism has come up because all of the chemicals are not names of real chemicals, but then have you ever seen a real monster?

The Incredible Labratory is intended for use by those in the third grade and older, with a reading level of at least the third grade. Adult supervision and instruction will be necessary at times. The program requires 48K, a disk drive, joystick is optional, and will run on all Atari computers.

— Robert Browning

SPARE CHANGE

(Broderbund Software, $32)

Occasionally, a software company will release a game that's a lot of fun to play, maybe a little strange, and hilarious to watch. If this kind of game interests you, then this is your disk.

As the owner of the Spare Change Arcade, you must keep your supply of game tokens in hand. But that's not always easy. Especially when two "Zerks" which escaped from a video game are out to steal all your tokens and put them in a piggy bank so they can retire.

You must move your animated man under a token machine and grab a token. Then you drop it in your token bin. If all the token machines run out, you can go snatch a dollar from the cash register and reload the machine. Sound simple? Not really, when those two Zerks are frantically checking machines, and taking them from your bin.

The Zerks will cooperate to keep you from taking back the token. They will pass it back and forth, and dodge your man in an effort to get the token to the piggy bank. They have a number of methods for depositing the token in the bank. Sometimes they will drop-kick it in, or just toss it. Fortunately, they are very poor shots, and miss frequently.

There are ways you can get back at them, though. There is a jukebox which will play a nice tune. While the Zerks dance wildly to this tune, you're collecting tokens! There are many other funny ways to delay the Zerks, but you'll have to find them.

The graphics and sound are very well done — especially the musical scores. Overall, this is a very good game, and I hope to see it on this year's best seller list.

— Tim Ebling

Review: **RAINY DAY ACTIVITIES FOR THE ATARI**

(Nancy Kozak Mayer, Ed.D., Reston, $12.95)

The author, Nancy Kozak, aptly describes **Rainy Day Activites for the Atari** as a book "designed to make the shortest possible program listings so that parents or older children would not have to spend so much time typing in the program that there would be no time left over for having fun." And, thus she has provided parents and young children, aged 3-9 years, with program listings (none over 20 lines long) for vidoegames, letter, word and number games, musical activities, and graphics.

The intent of this book is not to teach programming, but to make available over 50 simple program listing for children (and their parents) to successfully type and enjoy. Miles, my 5-year-old computer enthusiast son, was able to list the shorter programs with ease. The suggested variations for many of the programs gave him the opportunity to systematically change programs. This is a real plus, since he is inclined to list all programs he works with and randomly change listings just to see what will happen. Additionally, the book offers clear instructions including a "how it works", and "how it looks" section with each listing.

As user friendly as this book is, some skills in programming and knowledge about how one's computer works are needed. My knowledge of computers centers around using word processing programs, data base systems, and evaluation and use of programs created by other, more patient souls with programming skills. Luckily, Miles, my five-year-old, knew the Basic command for saving programs to disks, and had enough knowledge about program errors and proofreading and Basic editing commands to debug incorrectly listed programs. He patted his mother on the shoulder and said, "Mom, it's probably easier to list programs on rainy days."

Rainy Day Activities for the Atari is a recommended addition to the library of families who own Atari computers. It could be especially useful for families who have little programming background, and/or young children beginning to program. The book can be independently used by a 5-year-old (with good reading skills). It offers a variety of activities, including some interactive games for more than one child (or parent). Listings are short, and often program variations are provided. Of course, the ultimate test is. . . will your child use it. My son, Miles says he thinks other children will like and use **Rainy Day Activities**.

— Alice and Miles Erickson

PILOT'S MEASURE

by Carl Schwartz (ACE of Cleveland, 1/84)

Measure is a PILOT program designed for children. It provides practice in measuring or guessing the size of a line. While the 'lesson' part of this program is simple the reward can provide a large variety of designs. The turtle is told to DRAW,TURN, and repeat (the drawing) a certain number of times. The entire drawing routine is located on line 76.

Programming in PILOT is easier and faster than BASIC. If you have a child from five to twelve, or have some difficulty writing original BASIC programs, perhaps PILOT can help. Since PILOT uses more English type statements it can provide a user friendly environment for a beginner. For example, instead of using a number, PILOT's version of GOTO (called J:, for jump), tells the program to jump to the *LABEL (YOU give it a name). Also strings do not need dimensioning, text input is greatly simplified, and a LOGO like (moves according to his own current direction) turtle quickly draws your graphics. For the more experienced user large print screens and many of Atari BASIC's POKES (called Pointers) are supported.

A personal note: I am a special education teacher looking for good public domain software. Due to the special needs of the children in my school most commercial software is useless. Please write me at the Ashtabula County Board of Mental Retardation, 2506 South Ridge East, Ashtabula Ohio 44004, if you have a program we could use or you want a list of software we have found useful.

We have about 30 programs available for trading!! My students read (those who can) at about a preschool to 1st grade level and may have physical handicaps requiring special adaptive devices (for example a mounted joystick). These teenagers are not handicapped by a negative attitude, and once a job (or lesson) is learned they can do it well.

Sydney Brown: KONG

```
0 DIM P$(1):L=1:GOSUB 10000:LL=1:ML=3:
SC=0:HS=0
1 REM * ATARI COMPUTER ENTHUSISTS *
2 REM *      3662 VINE MAPLE DR      *
3 REM *        EUGENE, OR 97405      *
4 REM *              $12 YEAR        *
5 REM
6 REM ****KONG by Sydney Brown*****
7 REM              COMPCO
8 REM
9 REM             MARCH 1984
190 H=50:MC=1:V=97:CT=0:C=32:EN=1:D=1:
DN=0:X=6:Y=1:P$(V,V+11)=R1$:POKE 623,1
:M$=R1$:X1=6:Y1=1:E1=0:D1=1:DN1=0:C1=3
192 EX=0:POKE 53251,H:POKE 53248,81:PO
KE 53249,89
195 POSITION 15,0:? #6;"     ":POSITIO
N 15,0:? #6;SC;:IF ML>1 THEN FOR W=1 T
O ML-1:POSITION W-1,23:? #6;"■";:NEXT
W
200 ST=STICK(0):HD=0:IF ST<>15 THEN GO
SUB ST*10+1000:SOUND 0,35+10*MC,8,4:PO
KE 53278,0:IF CT=0 THEN P$(V,V+13)=M$
205 MC=MC+1:SOUND 0,0,0,0:IF MC>1 THEN
 MC=0
210 IF STRIG(0)=0 OR CT>0 THEN GOSUB 1
000
220 IF Y=Y1 AND ABS(X1-X)<2 THEN EN=1:
X=6:Y=1:DN=0:D=1
250 PP=PEEK(53255):IF PP=2 OR PP=3 OR
PP=6 OR PP=7 THEN GOSUB 2000
260 PP=PEEK(53263):IF PP=4 THEN 3000
270 IF MC=1 THEN 289
280 EX=EX+1:IF E1=1 THEN E1=0:GOSUB 40
0:X1=6
281 IF DN1>0 THEN 285
282 LOCATE X1+D1,Y1,Z:COLOR C1:PLOT X1
,Y1:IF Z=119 OR Z=120 THEN Z=32
283 X1=X1+D1:COLOR 119:PLOT X1,Y1:C1=Z
:IF X1=19 OR X1=0 OR X=INT(20*RND(0))
THEN D1=-D1:DN1=1:GOTO 287
284 GOTO 287
285 LOCATE X1,Y1+1,Z:COLOR C1:PLOT X1,
Y1:Y1=Y1+1:COLOR 128:PLOT X1,Y1:C1=Z:D
N1=DN1+1:IF DN1=4 THEN DN1=0
287 IF Y1=22 AND DN1>0 THEN COLOR C1:P
LOT X1,Y1:X1=6:Y1=1:E1=1:D1=1:DN1=0:C1
=32
288 GOTO 298
289 IF EX<80 THEN 298
290 IF EN=1 THEN EN=0:GOSUB 400:X=6:PO
KE 77,0
291 IF DN>0 THEN 295
292 LOCATE X+D,Y,Z:IF Z=119 OR Z=120 T
HEN Z=32
293 COLOR C:PLOT X,Y:X=X+D:COLOR 119:P
LOT X,Y:C=Z:IF X=19 OR X=0 OR X=INT(20
*RND(0)) THEN D=-D:DN=1:GOTO 297
294 GOTO 297
295 LOCATE X,Y+1,Z:COLOR C:PLOT X,Y:Y=
Y+1:COLOR 128:PLOT X,Y:C=Z:DN=DN+1:IF
DN=4 THEN DN=0
297 IF Y=22 AND DN>0 THEN COLOR C:PLOT
 X,Y:X=6:Y=1:EN=1:D=1:DN=0:C=32
298 IF EX>90 THEN EX=90
299 POKE 53250,128-X1:GOTO 200
400 POSITION 5,0:? #6;"i":POSITION 5,1
:? #6;"j":FOR W=1 TO 49:NEXT W
410 POSITION 5,0:? #6;"k":POSITION 5,1
:? #6;"lm":FOR W=1 TO 14:NEXT W:RETURN
999 GOTO 999
1000 IF CT=0 AND PEEK(53255)>1 THEN RE
TURN
1001 CT=CT+1:IF CT=5 THEN 1010
1002 P$(V,V+11)=J$:H=H+HD:IF CT=1 THEN
 SOUND 0,21,6,8
1005 RETURN
1010 CT=0:SOUND 0,0,0,0:IF HD=-1 THEN
P$(V,V+11)=L2$:RETURN
1019 P$(V,V+11)=R2$:RETURN
1050 GOTO 1070
1060 GOTO 1070
1070 IF (V-13)/12<>INT((V-13)/12) THEN
 RETURN
1072 IF H<200 THEN HD=1:H=H+4:POKE 532
51,H:IF MC=1 THEN M$=R1$:RETURN
1075 M$=R2$:RETURN
1090 GOTO 1110
1100 GOTO 1110
1110 IF V=5 THEN 1112
1111 IF (V-13)/12<>INT((V-13)/12) THEN
 RETURN
1112 IF V=13 AND H=144 THEN RETURN
1113 IF H>50 THEN HD=-1:H=H-4:POKE 532
51,H:IF MC=1 THEN M$=L1$:RETURN
1115 M$=L2$:RETURN
1130 RETURN
1140 IF CT>0 OR PEEK(53255)=0 THEN RET
URN
1141 IF PEEK(53255)<>4 THEN RETURN
1142 H=48+INT((H-44)/8)*8:POKE 53251,H
:P$(V,V+13)=C1$
1143 IF V>13 OR (H=144 AND V>5) THEN V
=V-2:SC=SC+5*LL:POSITION 15,0:? #6;SC
1145 IF MC=0 THEN M$=C1$:RETURN
1146 IF MC=1 THEN M$=C2$:RETURN
1149 RETURN
2000 ML=ML-1:FOR W=159 TO 0 STEP -7:SO
UND 0,49,8,INT(W/10):POKE 712,254*RND(
0):NEXT W:POKE 712,0
2010 P$(V,V+13)="◆◆◆◆◆◆◆◆◆◆◆◆◆◆":V=97:
H=52:POKE 53251,H:CT=0:P$(V,V+13)=R1$:
POKE 53278,0
2020 POSITION 0,0:? #6;"K":GOSUB 20100
:GOSUB 20100+100*L:GOSUB 20500
2040 IF ML>0 THEN 190
2099 GOTO 6000
3000 POKE 53250,128:RESTORE 32760:FOR
W=774 TO 783:READ A:POKE CB*256+W,A:NE
XT W
3010 POKE 53251,128:P$(5,16)=H$
3020 FOR KK=1 TO 2:FOR W=14 TO 7 STEP
-1:FOR WW=15 TO 0 STEP -1:SOUND 0,W,10
,WW:NEXT WW:NEXT W:NEXT KK
3050 COLOR 32:PLOT 5,0:PLOT 5,1
3051 FOR W=81 TO 217 STEP 8:POKE 53248
,W:POKE 53249,W+8:FOR A=15 TO 0 STEP -
1:SOUND 0,128,8,A:SOUND 1,122,8,A
3052 FOR B=1 TO 4:NEXT B:NEXT A:FOR A=
1 TO 14:NEXT A
3054 IF W<>105 THEN NEXT W:GOTO 3057
3055 FOR A=1 TO L:FOR Z=50 TO 100 STEP
 1.3:SOUND 0,Z,10,15:SOUND 0,150-Z,12,
10:NEXT Z:SOUND 0,0,0,0:FOR Z=1 TO 35
3056 NEXT Z:NEXT A:NEXT W
3057 FOR W=1 TO 100:NEXT W:FOR W=255 T
O 10 STEP -3:SOUND 0,W,10,4:NEXT W
3058 FOR W=1 TO LL*2:SC=SC+50:POSITION
 15,0:? #6;SC:FOR A=45 TO 0 STEP -1:SO
UND 0,7,10,A/3:NEXT A:NEXT W
3060 FOR W=1 TO 500:NEXT W:L=L+1:LL=LL
+1:IF L>3 THEN L=3
3990 POSITION 0,0:? #6;"K":GOSUB 20100
3998 GOSUB 20100+100*L:FOR W=774 TO 78
3:POKE CB*256+W,0:NEXT W:P$(5,V+11)="◆
◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆"
3999 GOSUB 20500:RESTORE 32060:FOR W=7
74 TO 783:READ A:POKE CB*256+W,A:NEXT
W:POKE 53278,0:GOTO 190
5000 POSITION 15,0:? #6;SC:POSITION 0,
2:IF SC>HS THEN HS=SC
5005 ? #6;"_A_A_A_A_A_A_A_A_A_AA      ■
SCORE=      __  TOP SCORE=      A";:PO
SITION 14,4:? #6;HS;:POSITION 14,3:? #
6;SC;
5990 POSITION 0,23:? #6;"_A_APRESS A_ST
ART_A":POSITION 0,0
5995 IF PEEK(53279)<>6 THEN POKE 711,I
NT(RND(0)*16)*16+12:GOTO 5995
5997 ? #6;"K":GOSUB 20100:GOSUB 20500
5998 GOSUB 20200:FOR W=774 TO 783:POKE
 CB*256+W,0:NEXT W:P$(5,V+11)="◆◆◆◆◆◆◆
◆◆◆◆◆◆◆◆◆◆◆"
5999 L=1:ML=3:SC=0:RESTORE 32060:FOR W
=774 TO 783:READ A:POKE CB*256+W,A:NEX
T W:POKE 53278,0:GOTO 190
6000 FOR W=160 TO 0 STEP -10:FOR WW=W
TO W+20:SOUND 0,WW,10,10:NEXT WW:NEXT
W:SOUND 0,0,0,0:LL=1:L=1:GOTO 5000
10000 DIM M$(14),C1$(14),C2$(14),L1$(1
2),L2$(12),R1$(12),R2$(12),J$(12),W$(1
```

```
2)
10001 CB=PEEK(106)-4:POKE 106,CB:GOSUB
 20000:A=CB*256:FOR B=0 TO 511
10002 IF B>431 OR B<327 AND B<360 THEN
 READ D:POKE A+B,D:GOTO 10010
10005 POKE A+B,PEEK(57344+B)
10010 NEXT B:POKE 54279,CB
10020 FOR M=CB*256+512 TO CB*256+1024:
POKE M,0:NEXT M:POKE 53256,0:POKE 5325
7,0
10030 POKE 53248,81:POKE 53249,89:POKE
 53250,126:POKE 53251,52
10040 FOR M=521 TO 536:READ A,B:POKE C
B*256+M,A:POKE CB*256+M+128,B:NEXT M
10041 FOR M=774 TO 783:READ A:POKE CB*
256+M,A:NEXT M
10050 C1$="[illegible]":C2$="[illegible]":L1$="[illegible]":L2$="[illegible]"
10051 R1$="[illegible]":R2$="[illegible]":J$="[illegible]":H$="[illegible]"
10060 VT=PEEK(134)+256*PEEK(135):AT=PE
EK(140)+256*PEEK(141):OF=256*CB+896-AT
:V3=INT(OF/256):V2=OF-256*V3
10061 POKE VT+2,V2:POKE VT+3,V3:POKE V
T+4,128:POKE VT+5,0:POKE VT+6,128:POKE
 VT+7,0
10100 POSITION 0,10:? #6;"[illegible]
10110 IF PEEK(53279)<>6 THEN 10110
10120 SOUND 0,0,0,0:SOUND 2,0,0,0
10500 GRAPHICS 17:POKE 708,136:POKE 70
9,26:POKE 710,222:POKE 711,90:POKE 712
,0:POKE 756,CB
10510 POKE 559,46:POKE 53277,3:POKE 70
4,40:POKE 705,40:POKE 706,78:POKE 707,
14
10600 GOSUB 20100:GOSUB 20100+100*L:GO
SUB 20500
10999 RETURN
20000 GRAPHICS 18:POKE 710,220:POKE 71
2,18:FOR M=0 TO 10 STEP 2:POSITION 7,M
:? #6;"K O N G":NEXT M
20010 SOUND 0,0,0,0:POKE 53768,7:POKE
53765,168:POKE 53760,254:POKE 53761,16
8:POKE 53764,127
20099 RETURN
20100 FOR M=2 TO 20 STEP 3:POSITION 0,
M:? #6;"_A_A_A_A_A_A_A_A_A_A";:NEXT M:
POSITION 0,23:? #6;"_A_A_A a.c.e.A_A_A
"
20110 COLOR 94:PLOT 19,23:POSITION 7,0
:? #6;"A_A_A[illegible]":POSITION 7,1:? #6;"_
 [illegible]"
20111 POSITION 19,2:? #6;"[illegible]":POSITION
19,3:? #6;"[illegible]":POSITION 19,4:? #6;"[illegible]"
20115 POSITION 5,0:? #6;"k":POSITION 5
,1:? #6;"l"
20120 POSITION 0,0:? #6;"vv":POSITION
0,1:? #6;"vvv"
20199 RETURN
20200 FOR M=8 TO 20 STEP 6:IF M<20 THE
N X=0:POSITION X,M:? #6;"[illegible]":POSITION X
,M+1:? #6;"[illegible]":POSITION X,M+2:? #6;"[illegible]"
20205 X=19:POSITION X,M:? #6;"[illegible]":POSIT
ION X,M+1:? #6;"[illegible]":POSITION X,M+2:? #6
;"[illegible]":NEXT M
20210 FOR M=5 TO 20 STEP 6:X=10:POSITI
ON X,M:? #6;"[illegible]":POSITION X,M+1:? #6;"[illegible]
":POSITION X,M+2:? #6;"[illegible]":NEXT M
20220 POSITION 2,7:? #6;"[illegible] z z[illegible] y
z y":POSITION 2,13:? #6;"z [illegible] y [illegible] y
  z":POSITION 3,19:? #6;"z y [illegible]z y"
20299 RETURN
20300 FOR M=8 TO 20 STEP 6:X=16:POSITI
ON X,M:? #6;"[illegible]":POSITION X,M+1:? #6;"[illegible]
":POSITION X,M+2:? #6;"[illegible]":NEXT M
20310 FOR M=5 TO 20 STEP 6:X=3:POSITIO
N X,M:? #6;"[illegible]":POSITION X,M+1:? #6;"[illegible]
":POSITION X,M+2:? #6;"[illegible]":NEXT M
20320 POSITION 4,7:? #6;"y z y [illegible] y [illegible]z
y":POSITION 4,13:? #6;"y [illegible] z [illegible]y z z"
:POSITION 4,19:? #6;"z y [illegible]y [illegible]z"
20399 RETURN
20400 FOR M=2 TO 20 STEP 6:X=19:POSITI
ON X,M:? #6;"[illegible]":POSITION X,M+1:? #6;"[illegible]
":POSITION X,M+2:? #6;"[illegible]":NEXT M
20410 FOR M=5 TO 20 STEP 6:X=0:POSITIO
N X,M:? #6;"[illegible]":POSITION X,M+1:? #6;"[illegible]"
:POSITION X,M+2:? #6;"[illegible]":NEXT M
20420 POSITION 2,7:? #6;"zz [illegible] y yy z
 yy":POSITION 3,13:? #6;"z yy  zz y [illegible]
[illegible]":POSITION 3,19:? #6;"[illegible]zz y [illegible] zy [illegible]z
"
20499 RETURN
20500 FOR M=4 TO 16 STEP 6:FOR MM=1 TO
 7
20510 X=INT(RND(0)*20):LOCATE X,M,Z:IF
 Z<>32 THEN 20510
20520 POSITION X,M:? #6;"[illegible]":NEXT MM:NE
XT M:RETURN
32000 DATA 16,56,56,120,112,112,224,22
4,196,202,213,213,245,213,74,52
32005 DATA 16,56,56,56,28,28,12,12,14,
14,6,6,7,6,2,3
32010 DATA 255,255,255,255,255,255,255
,255,24,36,90,90,90,90,36,24
32015 DATA 60,255,195,255,195,255,195,
60,24,36,90,90,90,90,36,24,12,12,0,54,
54,0,219,219
32020 DATA 0,0,0,0,0,0,0,255,129,255,1
29,129,129,255,129,129,56,16,254,186,1
86,40,40,108
32030 DATA 255,255,48,24,12,6,255,255,
255,255,6,12,24,48,255,255
32050 DATA 0,0,0,6,0,15,0,62,0,255,3,2
53,7,254,15,252,29,120,31,176,31,192,3
1,192,248,192,240,224,192,240,32,32
32060 DATA 28,93,73,127,28,62,127,20,2
0,54
32760 DATA 28,62,127,127,63,31,15,7,3,
1,0,0
```

Steve Monn:
THE GREEN MACHINE

```
1 REM FROM Frederick ALE, Dec 1983
2 REM BY Steve Monn
10 REM GREEN SCREEN LISTING 1
100 ? CHR$(125)
110 ? "ENTER THE SCREEN COLOR DESIRED
":? "USE THE COLORNUMBER, PLEASE":TRAP
 110:INPUT COLOR
120 IF COLOR<0 OR COLOR>15 THEN ? "ENT
ER A COLOR FROM 0 TO 15, PLEASE.":GOTO
 110
130 ? CHR$(125)
140 ? "ENTER THE NUMBER OF FLASHES PER
 SECONDFOR THE CURSOR ";:TRAP 140:INPU
T FPS
150 IF FPS<1 OR FPS>60 THEN ? "ENTER A
 VALUE FROM 1 TO 60, PLEASE.":GOTO 140
160 FPS=INT(60/FPS)
170 GOSUB 1070
180 POKE 1654,COLOR*16
190 POKE 1577,FPS
200 END
1070 FOR I=1536 TO 1536+45
1080 READ B:POKE I,B:NEXT I
1090 A=USR(1536)
1100 RETURN
1110 DATA 104,169,17,141,40,2,169,6
1120 DATA 141,41,2,169,30,141,26,2
1130 DATA 96,169,208,141,198,2,141,200
1140 DATA 2,169,214,141,197,2,173,243
1150 DATA 2,41,2,73,2,141,243,2
1160 DATA 169,10,141,26,2,96
```

Listing 2:

```
*=$0600
PLA
LDA #
STA $0228
LDA $06
STA $0229
LDA $1C
STA $021A
RTS
---------
LDA $D0
STA $02C6
STA $02C8
LDA $D6
STA $02C5
LDA $02F3
AND #$02
EOR #$02
STA $021A
LDA #$0F
STA $021A
RTS
```

Carl Schwartz:MEASURE

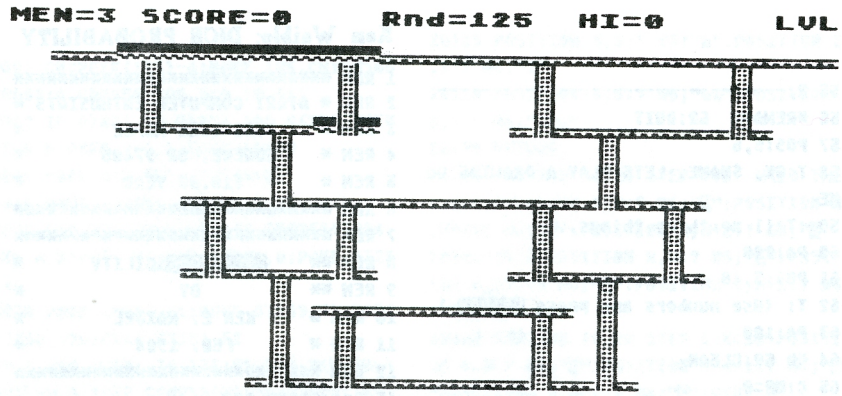
PILOT program from Cleveland ACE

```
1 PO5:5,5
2 T:Hi, I'M a measuring program..
3 T:
4 T:Who are you ?  \
5 A:$NAME
6 PO5:5,15
7 T:Ok, $NAME lets go...
8 :Pick the length of my line in inche
s. (and press RETURN)
9 PA(#R(1)  :220
10 R:--------------------------------
11 *MEASUREOVER  VNEW#
12 R:--------------------------------
13 *MEASURE E(#Q)0):
14 C:#M=?\6+1
15 C:#O=#M*15
16 R:---------- Adjust the number(15)
if your TV is not a 13in ------------
--
17 GR:CLEAR
18 GR:TURNTO 90                 [horiz
19 C:#V=?\7*10-15
20 GR:TURNTO90;GOTO-65,#V ;DRAW #O
21 GR:TURNTO90;GOTO-65,#V+1 ;DRAW #O
22 *REDO E(#Q)1):
23 , T:ROLL DICE (1 2 3 4 5 or 6)
24 , A:#A
25 , M:#M
26 , C(#A()#D):#M=#M+1
27 , JM:*TRYMORE
28 , J(#T)4 ):*DONE [no of ?
29 , #Y:*GOOD
30 , J(#T(11):*MEASURE
31 , E:
32 *TRYMORE  T:TRY AGAIN $NAME
33 , PA:30
34 , J:*REDO
35 R:--------------------------------
36 *GOOD
37 T:GOOD ANSWER
38 C:#T=#T+1
39 C:#5=0
40 *LOOP SO:#5
41 ,    C:#5=#5+1
42 ,    J(#5(31):*LOOP
43 SO:
44 E:
45 R:--------------------------------
46 *DONE SO:#Z
47 C:#Z=(10-(2*#M#D)*10
48 T:$NAME, MISSED #M and got #Z %RIG
HT \
49 PA:150
50 J(#M#D)0):*REPEAT
51 T(#M=0): YOU REALLY MEASURE UP $NA
ME
52 J(#M=0):*REWARD
53 C:#Q=#Q+1
54 E:
55 R:--------------------------------
56 *REWARD  GR:QUIT
57 PO5:5,5
58 T:OK, $NAME, LETS PLAY A DRAWING GA
ME
59 :Tell me three things...
60 PA:220
61 PO5:5,18
62 T: (Use numbers and press RETURN )
63 PA:100
64 *D GR:CLEAR
65 C:#P=0
66 T:Length to draw (1-100)
67 A:#D
68 GR:DRAW#D
69 T:Turn  Angle (1-90)
70 A:#T
71 GR:GOTO 0,0;TURNTO #T;DRAW 25;TURN
-135;DRAW 5;TURN180;DRAW5;TURN90;DRAW5
72 T:Number of trips (10-200)
73 A:#R
74 GR:CLEAR; GOTO 0,0;TURNTO 0
75 *REPETITION   SO:#C,#P
76 , C:#C=?\3     [random colors
77 , GR:PENYELLOW
78 , GR(#C=1):PEN BLUE
79 , GR(#C=2):PEN RED
80 , GR:DRAW#D;TURN180;GO#T;TURN270+#T
;GO 1
81 , C:#P=#P+1
82 , J(#R)#P):*REPETITION
83 SO:0,0
84 T:$NAME'STURTLE #TTURN #DLONG #RTRI
P\
85 R:--------------------------------
86 *REPEAT PA:60
87 SO:
88 T:  Again ?(Y,N)\
89 A:
90 M:Y
91 JM:*MEASUREOVER
92 T:So long, $NAME.....
93 PA:120
94 E:
95 R:--------------------------------
96 R:by      Carl Schwartz 5/83
97 R:  Designed for pre-schoolers
98 R: adjust line LENGTH for your TV
99 R:PO.BOX 52, KINGSVILLE, OHIO 44048
100 R:-------------------------------
```

Ken Waible: DICE PROBABILITY

```
1 REM ******************************
2 REM * ATARI COMPUTER ENTHUSIATS *
3 REM *    3662 VINE MAPLE        *
4 REM *    EUGENE, OR 97405       *
5 REM *      $10.00 YEAR          *
6 REM ******************************
7 REM ******************************
8 REM **    DICE PROBABILITY      *
9 REM **          BY              *
10 REM *     KEN E. WAIBEL         *
11 REM *       FEB. 1984           *
12 REM ******************************
15 C=1:GOSUB 600
20 CLR
30 ? "%":REM Number of rolls
32 ? :? :? "Display individual dice ro
lls(yes/no)";:GOSUB 500
34 ? :? :? "How many dice rolls";
36 INPUT NUM:NUMBER=NUM+NUMBER
37 IF R=89 THEN POKE 201,2
38 FOR X=1 TO NUM
40 REM ROLL DICE
42 DICE1=INT(RND(0)*6+1)
44 DICE2=INT(RND(0)*6+1)
46 TOTAL=DICE1+DICE2
50 REM Adds the die
54 IF R=89 THEN PRINT TOTAL,
56 GOSUB TOTAL+60
58 NEXT X
59 GOTO 100
60 REM COUNT DICE
62 T2=T2+1:RETURN :REM TOTAL=2
63 T3=T3+1:RETURN :REM TOTAL=3
64 T4=T4+1:RETURN :REM Etc. etc.
65 T5=T5+1:RETURN
66 T6=T6+1:RETURN
67 T7=T7+1:RETURN
68 T8=T8+1:RETURN
69 T9=T9+1:RETURN
70 T10=T10+1:RETURN
71 T11=T11+1:RETURN
72 T12=T12+1:RETURN
100 REM PRINTING
102 C=3:GOSUB 600
105 POKE 201,10
110 ? :? "What now?"
112 ? "     1-Print to screen"
114 ? "     2-Print to printer"
116 ? "     3-Add more dice rolls"
118 ? "     4-Start over"
120 ? "     5-End Program"
122 GOSUB 500
124 IF R=49 THEN OPEN #2,8,0,"S:":GOTO
 150
126 IF R=50 THEN OPEN #2,8,0,"P:":GOTO
 150
```

GOTO Page 12



Stan Ockers: RATS REVENGE

```
; RATS REVENGE IN ACTION!
; by Stan Ockers

; ACE Newsletter, 3662 Vine Maple
; Eugene, OR 97405  $12 year
;          March 1984

MODULE ; RATS' REVENGE
  CARD pabase,dest,source,cnt,source1,
       source2,source3,score,maxscore,
       rndpts,dlist=560
  BYTE papage,stk,v,w,col,row,
       vcount=$D40B,consol=53279,dir,all,
       dir1,blkflg,dir2,
       dir3,pos,pos1,pos2,pos3,
       level,dotscnt,hitflg,men,ownflg
  BYTE ARRAY ratup=[0 66 126 60 24 66
    126 90 126 60 36 102 0 0],
    ratrt=[0 32 56 100 62 16 38 126 120
    120 120 44 48 0],ratlt=[0 4 28 54
    124 8 92 126 30 30 28 32 12 0],
    cursor=[34 20 8 20 34],bugup=[0 36
    24 60 126 60 24 60 90 60 24 60
    0 0],bugrt=[0 48 32 120 100
    56 16 56 56 60 56 44 48 0],
    buglt=[0 12 4 30 54 28 8 20 20 60
    20 52 12 0],allowed(132),saveall(133),
    hortop=[3 32 32 32], vertop=[3 32
    6 32],allbot=[3 7 6 7],newall=[3
    135 6 135],horbot=[3 7 7 7],
    dots(132),blk=[3 32 32 32],
    newbot=[3 135 135 135]
  INT delx,dely,delx1,dely1,delx2,
      dely2,delx3,dely3,x,y,x1,y1,
      x2,y2,x3,y3,ax,ay
PROC Init()
  SetBlock(allowed,132,0)
  SetBlock(dots,132,1)
  FOR cnt=1 TO 10
  DO allowed(cnt)=76 OD
  allowed(0)=72 allowed(11)=68
RETURN

PROC Painit()
  BYTE ARRAY fil=[65 85 65 85 65 85 65
    85 0 0 0 0 255 51 204 255 0 0 255 255
    0 24 60 60 24 0 255 255 0 0 0 0
    0 0 255 255 0 0 0]
  papage=Peek(106)-16 pabase=papage*256
  Poke(54279,papage) Poke(559,62)
  Poke(53277,3) Poke(53248,x) Poke(704,92) RETURN
  Poke(705,202) Poke(706,250) Poke(707,186)
  SetBlock(pabase+1024,1024,0)
  Moveblock(pabase,57344,1024)
  Poke(756,papage)
  FOR cnt=0 TO 39
  DO Poke(pabase+cnt+560,fil(cnt)) OD
  FOR cnt=6 TO 28
  DO Poke(dlist+cnt,4) OD
RETURN

PROC Ding(BYTE pitch,CARD dly)
  BYTE loud
  CARD wait
   FOR loud=0 TO 15
   DO Sound(0,pitch,10,15-loud)
     FOR wait=1 to dly DO OD OD
   Sndrst()
RETURN

PROC Tracks() ; leave trail
  BYTE newpos,z
 IF newpos<>pos THEN
  newpos=pos v=pos MOD 12 w=pos/12
  z=Locate(3*v+2,2*w+2)
  IF z=7 THEN
    rndpts==+25 Position(21,0)
    PrintC(rndpts)
    Position(3*v+2,2*w+2) dotscnt==-1
    IF allowed(pos)&64 THEN Print(newbot)
    ELSEIF allowed(pos)&16 THEN
        Print(newall) FI FI FI
RETURN
PROC Move() ; move the bug
  stk=Stick(0) ! 15
  IF (y-10) MOD 16=0 AND (x-22) MOD 12=0 THEN
    row=(y-42)/16 col=(x-56)/12
    pos=12*row+col all=allowed(pos)&15
    IF stk=8 AND dir=4 THEN dir=8
    ELSEIF stk=4 AND dir=8 THEN dir=4
    ELSEIF stk=1 AND dir=2 THEN dir=1
    ELSEIF stk=2 AND dir=1 THEN dir=2
    FI
    IF (dir & 3)>0 THEN stk==& 12
    ELSE stk==& 3 FI
    IF(stk & all)>0 THEN dir=stk FI
    WHILE (dir & all)=0
    DO dir=dir RSH 1
       IF dir=0 THEN dir=8 FI OD
    IF dir=1 THEN dely=-1 delx=0
          source=bugup
    ELSEIF dir=2 THEN dely=1 delx=0
          source=bugup
    ELSEIF dir=4 THEN dely=0 delx=-1
          source=buglt
    ELSEIF dir=8 THEN dely=0 delx=1
          source=bugrt FI FI
    x==+delx y==+dely
PROC Move1() ; decide rat #1 movement
  BYTE nua,col1,row1
  IF (y1-10) MOD 16=0 AND (x1-22) MOD 12=0 THEN
    row1=(y1-42)/16 col1=(x1-56)/12
    pos1=12*row1+col1 all=allowed(pos1)
    IF Rand(15)>10-level THEN
      ax=x-x1 IF ax<0 THEN ax=-ax FI
      ay=y-y1 IF ay<0 THEN ay=-ay FI
      IF ax>ay THEN
        IF (x1>x) THEN dir1=4
        ELSEIF (x1<x) THEN dir1=8 FI
      ELSE
        IF (y1<y) THEN dir1=2
        ELSEIF (y1>y) THEN dir1=1 FI FI
    FI
    WHILE (dir1 & all)=0
    DO dir1=dir1 RSH 1
      IF dir1=0 THEN dir1=8 FI OD
    IF dir1=1 THEN dely1=-1 delx1=0
          source1=ratup
    ELSEIF dir1=2 THEN dely1=1 delx1=0
          source1=ratup
    ELSEIF dir1=4 THEN dely1=0 delx1=-1
          source1=ratlt
    ELSEIF dir1=8 THEN dely1=0 delx1=1
          source1=ratrt FI FI
RETURN

PROC Move2()
  BYTE nua,col2,row2
  IF (y2-10) MOD 16=0 AND (x2-22) MOD 12=0 THEN
    row2=(y2-42)/16 col2=(x2-56)/12
    pos2=12*row2+col2 all=allowed(pos2)
```

```
 IF Rand(15)>16-level THEN
      ax=x-x2 IF ax<0 THEN ax=-ax FI
      ay=y-y2 IF ay<0 THEN ay=-ay FI
      IF ax>ay THEN
        IF (x2>x) THEN dir2=4
        ELSEIF (x2<x) THEN dir2=8 FI
      ELSE
        IF (y2<y) THEN dir2=2
        ELSEIF (y2>y) THEN dir2=1 FI FI
    FI
    WHILE (dir2 & all)=0
    DO dir2=dir2 RSH 1
       IF dir2=0 THEN dir2=8 FI OD
    IF dir2=1 THEN dely2=-1 delx2=0
         source2=ratup
    ELSEIF dir2=2 THEN dely2=1 delx2=0
         source2=ratup
    ELSEIF dir2=4 THEN dely2=0 delx2=-1
         source2=ratlt
    ELSEIF dir2=8 THEN dely2=0 delx2=1
         source2=ratrt FI FI
RETURN

PROC Move3()
  BYTE num,col3,row3
  IF (y3-16) MOD 16=0 AND (x3-22) MOD 12=0 THEN
    row3=(y3-42)/16 col3=(x3-56)/12
    pos3=12*row3+col3 all=allowed(pos3)
    IF Rand(15)>16-level THEN
      ax=x-x3 IF ax<0 THEN ax=-ax FI
      ay=y-y3 IF ay<0 THEN ay=-ay FI
      IF ax>ay THEN
        IF (x3>x) THEN dir3=4
        ELSEIF (x3<x) THEN dir3=8 FI
      ELSE
        IF (y3<y) THEN dir3=2
        ELSEIF (y3>y) THEN dir3=1 FI FI
    FI
    WHILE (dir3 & all)=0
    DO dir3=dir3 RSH 1
       IF dir3=0 THEN dir3=8 FI OD
    IF dir3=1 THEN dely3=-1 delx3=0
         source3=ratup
    ELSEIF dir3=2 THEN dely3=1 delx3=0
         source3=ratup
    ELSEIF dir3=4 THEN dely3=0 delx3=-1
         source3=ratlt
    ELSEIF dir3=8 THEN dely3=0 delx3=1
         source3=ratrt FI FI
RETURN

PROC Update() ; all players move
  DO UNTIL vcount=128 OD
  Poke(53248,x) dest=pmbase+1024+y
  MoveBlock(dest,source,14)
  x1==+delx1 y1==+dely1 Poke(53249,x1)
  dest=pmbase+1280+y1
  MoveBlock(dest,source1,14)

PROC Bkgd() ; draw ladders & girders
  Graphics(0) Poke(752,1)
  FOR pos=0 TO 131
  DO x=pos MOD 12 y=pos/12
    all=allowed(pos)
    Position(3*x+2,2*y+1)
    IF all&48 THEN Print(vertop)
    ELSE Print(blk) FI
    Position(3*x+2,2*y+2)
    IF all&64 THEN Print(horbot)
    ELSEIF all&32 THEN Print(vertop)
    ELSEIF all&16 THEN Print(allbot)
    ELSE Print(blk) FI OD
  FOR x=1 TO 38
  DO Position(x,23) Put(13) OD
RETURN

PROC Dly(CARD maxcnt)
  CARD cnt
  FOR cnt=1 to maxcnt DO OD
RETURN

PROC Saveaze()
  Open(1,"D:MAZE",8,0)
  FOR pos=0 TO 132
  DO saveall(pos+1)=allowed(pos) OD
  saveall(0)=132 PrintD(1,saveall)
  Close(1)
RETURN

PROC Cleanscr() ; get rid of players
  FOR cnt=53248 TO 53251
    DO Poke(cnt,0) OD
RETURN

PROC Openup() ; pos. above permitted
  IF allowed(pos-12) & 112 THEN
  allowed(pos-12)==%2 allowed(pos)==%1
  FI RETURN

PROC Opendn()
  IF allowed(pos+12) & 48 THEN
  allowed(pos+12)==%1 allowed(pos)==%2
  FI RETURN

PROC Openlt()
  IF allowed(pos-1) & 80 THEN
  allowed(pos-1)==%8 allowed(pos)==%4
  FI RETURN

  x2==+delx2 y2==+dely2 Poke(53250,x2)
  dest=pmbase+1536+y2
  MoveBlock(dest,source2,14)
  x3==+delx3 y3==+dely3 Poke(53251,x3)
  dest=pmbase+1792+y3
  MoveBlock(dest,source3,14) Tracks()
RETURN

PROC Openrt()
  IF allowed(pos+1) & 80 THEN
  allowed(pos+1)==%4 allowed(pos)==%8
  FI RETURN

PROC Setbar() ; create screen
  BYTE key
  x=70 y=61 source=cursor
  dest=pmbase+1024+y Poke(764,255)
  SetBlock(pmbase+1024,256,0)
  Position(0,0)
  Print("(1) (2)_(3)|(4)^(5)Save")
  Print("(6)Exit (arrows)")
PROC Rats()
  DO
  Poke(53248,x) dest=pmbase+1024+y
  MoveBlock(dest,source,5)
   DO key=Peek(764)
    UNTIL key<>255 OD
  SetBlock(dest,5,0) Poke(764,255)
  IF key=15 AND y<181 THEN Y==+16
  ELSEIF key=14 AND y>61 THEN y==-16
  ELSEIF key=7 AND x<178 THEN x==+12
  ELSEIF key=6 AND x>70 THEN x==-12
  FI
  IF consol=5 THEN EXIT FI
  col=(x-58)/12 row=(y-45)/16
  pos=12*row+col
  Position(3*col+2,2*row+1)
  IF key=31 THEN
  Print(blk) allowed(pos)=0
    allowed(pos-1)==&247
    allowed(pos+1)==&251
    allowed(pos-12)==&253
    allowed(pos+12)==&254
  ELSEIF key=30 THEN
    allowed(pos)=64 allowed(pos-12)==&253
    Openrt() Openlt() Opendn()
    Print(hortop)
  ELSEIF (key=26) THEN
    allowed(pos)=32 allowed(pos-1)==&247
    allowed(pos+1)==&251 Openup()
    Opendn() Print(vertop)
  ELSEIF (key=24) THEN
    allowed(pos)=16 Openup() Opendn()
    Openlt() Openrt() Print(vertop)
  FI
  Position(3*col+2,2*row+2)
  IF key=31 THEN Print(blk)
  ELSEIF key=30 THEN Print(horbot)
  ELSEIF key=26 THEN Print(vertop)
  ELSEIF key=24 THEN Print(allbot)
  FI
  IF key=29 THEN Saveaze() FI
  IF key=27 THEN EXIT FI
  DO UNTIL PEEK(764)<>key OD
OD
RETURN
```

```
PROC Loadmaze()
  Open(1,"D:MAZE",4,0) InputSD(1,saveall)
  SCopyS(allowed,saveall,2,132)
  allowed(0)=saveall(1) Close(1)
RETURN

PROC Contact() ; hit a rat
  FOR v=50 to 200 STEP 5
    DO Ding(v,v) OD
  hitflg=1
RETURN

PROC Rndover() ; completed screen
  FOR v=50 to 200 STEP 5
    DO Ding(250-v,v) OD
RETURN

PROC Gameover()
  Graphics(18) Position(6,4)
  PrintD(6,"GAME OVER")
  Position (0,9)
  PrintD(6,"press START to play")
  DO UNTIL consol=6 OD
  DO UNTIL (consol&7)=7 OD RETURN

PROC Scorecard()
  Position(0,0) Print("MEN=") PrintB(men)
  Position(6,0) Print("SCORE=")
  PrintC(score) Position(17,0)
  Print("Rnd=") PrintC(rndpts)
  Position(26,0) Print("HI=")
  PrintC(maxscore) Position(35,0)
  Print("LVL=") PrintB(level)
RETURN

PROC Play()
BYTE ARRAY lvl1=[72 78 76 76 78 76 76
78 76 76 78 68 0 35 0 0 35 0 0 35 0 0
35 0 0 25 76 78 21 0 0 25 78 76 21 0
0 0 0 35 0 0 0 0 35 0 0 0 0 0 74 29 78
76 76 78 29 78 0 0 0 0 35 0 35 0 0 35
0 35 0 0 0 0 25 78 21 0 0 25 78 21 0 0
0 0 0 35 74 76 76 78 35 0 0 0 0 0 0 35
35 0 0 35 35 0 0 0 0 0 0 25 29 76 76
29 21 0 0 0 0 0 0 0 0 0 0 0 0 0 0],
lvl4=[72 78 76 76 78 76 78 76 76 76 78
68 0 25 76 76 23 0 27 76 76 76 21 0 0
0 0 0 27 76 23 0 0 0 0 0 0 74 76 76 31
76 31 76 76 76 78 0 0 25 76 76 23 0 27
76 76 76 21 0 0 0 0 0 0 27 76 23 0 0 0 0
0 0 74 76 76 31 76 31 76 76 76 78 0 0
25 76 76 23 0 27 76 76 76 21 0 0 0 0 0
27 76 23 0 0 0 0 0 0 0 0 72 29 76 29
68 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0],
lvl7=[72 78 76 76 78 76 78 78 76 78 76 78
68 0 25 78 29 78 21 35 0 35 0 35 0 0
74 29 78 29 78 29 78 29 78 21 0 0 35
0 35 0 35 74 29 78 29 78 0 0 25 78 29
78 21 35 0 35 0 35 0 0 74 29 78 29 78
29 78 29 78 21 0 0 35 0 35 0 35 74 29
78 29 78 0 0 25 78 29 78 21 35 0 35 0
35 0 0 0 35 0 35 0 25 78 29 78 21 0
0 72 29 76 29 76 76 29 76 29 68 0 0 0
0 0 0 0 0 0 0 0 0 0 0]
  DO
    IF ownflg=0 THEN
      IF level>0 AND level<4 THEN
      FOR v=0 TO 132
      DO allowed(v)=lvl1(v) OD
      ELSEIF level>3 AND level<7 THEN
      FOR v=0 TO 132
      DO allowed(v)=lvl4(v) OD
      ELSEIF level>6 THEN
      FOR v=0 TO 132
      DO allowed(v)=lvl7(v) OD FI FI
    Bkgd() Painit() Scorecard()
    x=56 y=42 dir=8 source=ratrt
    SetBlock(pabase+1024,1024,0)
    y1=42 dir1=4 source1=ratlt
    source2=ratlt
    y2=42 dir2=4 delx=1
    y3=42 dir3=4 source3=ratlt
    pos=0 pos1=11 pos2=11 pos3=11
    dely=0 dely1=0 dely2=0 dely3=0
    delx1=-1 delx2=0 delx3=0 x1=184
    x2=0 x3=0 dotscnt=0
    FOR cnt=0 to 131
      DO w=allowed(cnt)
        IF w<>0 THEN dotscnt=+1 FI OD
    w=(level-1)MOD 3
    IF w>0 THEN x2=184 delx2=-1 FI
    IF w>1 THEN x3=184 delx3=-1 FI
    DO FOR cnt=1 TO 10-level
        DO DO UNTIL vcount=1 OD
        UNTIL vcount=128 OD
    Move() Move1() Move2()
    Move3() Update()
    IF dotscnt=0 THEN Rndover()
      EXIT FI
    IF((x=x1)AND(y=y1))OR
      ((x=x2)AND(y=y2))OR
      ((x=x3)AND(y=y3)) THEN
      Contact() EXIT FI
    Poke(77,0)
    OD
  Cleanscr()
  IF dotscnt=0 THEN level=+1
    IF level=10 THEN level=9 FI
    score=+rndpts rndpts=0
    IF maxscore<score THEN
      maxscore=score FI
  ELSEIF hitflg=1 THEN men=-1
    rndpts=0
    IF men=0 THEN Gameover() EXIT FI
  FI
  OD
RETURN

PROC Rats()
  DO
    Graphics(18) Position(4,1)
    PrintD(6,"RAT'S REVENGE")
    Position(6,3) PrintDE(6," 1 REGULAR"
    PrintD(6,"option   CREATE")
    Position(9,5) PrintDE(6,"PLAY OWN")
    Position(0,7)
    PrintD(6,"select LVL 123456789")
    Position(0,9) PrintD(6,"start")
    Position(1,11) Poke(dlist+16,6)
    PrintD(6,"WRITTEN IN ACTION")
    w=3 level=1 ownflg=0 score=0
    maxscore=0 rndpts=0 men=3
    DO
    DO UNTIL(consol&7)<>7 OD
    IF consol=3 THEN position(7,w)
      PutD(6,32) w=+1
      IF w=6 THEN w=3 FI
      Position(7,w) PutD(6,178)
      Ding(121,600)
    ELSEIF consol=5 THEN
      Position(18+level,7)
      PutD(6,48+level) level=+1
      IF level=10 THEN level=1 FI
      Position(18+level,7)
      PutD(6,176+level) Ding(60,600)
    ELSEIF consol=6 THEN
      IF w=4 THEN Init() Bkgd()
        Painit() Setbar() Cleanscr()
        EXIT
      ELSEIF w=5 THEN Loadmaze()
        ownflg=1 Play() ownflg=0
        Cleanscr() EXIT
      ELSEIF w=3 THEN
        Play() Cleanscr() EXIT
      FI
    FI
    DO UNTIL(consol&7)=7 OD OD OD
RETURN
```

Meeting

Weds March 14, 7:30
South Eugene High
Cafeteria

C. Mueller: String Search Routine

```
0010          .OR $0600
0020 ADTOT    .EQ $CB      ;ADR(TOT$)
0030 ENDTOT   .EQ $CD      ;END TOT$
0040 ADSS     .EQ $D6      ;ADR(SS$)
0050 LENSS    .EQ $D8      ;LEN(SS$)
0060 POSIT    .EQ $D4      ;POSITION
0061 ;
0062 ;PULL ARGUMT PASSED BY USR
0063 ;
0070          PLA          ;# ARGUMTS
0080          PLA          ;MSB/ADR(TOT$)
0090          STA ADTOT+1
0100          PLA          ;LSB/ADR(TOT$)
0110          STA ADTOT+1 ;
0120          PLA          ;MSB/LEN(TOT$)
0130          CLC          ;CLR CARRY FLAG
0140          ADC ADTOT+1 ;ADD THE ADR(TOT$)
0150          STA ENDTOT+1
0160          PLA
0170          ADC ADTOT
0180          STA ENDTOT
0190          PLA          ;MSB/ADR(SS$)
0200          STA ADSS+1
0210          PLA          ;LSB/ADR(SS$)
0220 STA ADSS
0230          PLA          ;THROW AWAY MSB OF LEN(SS$)
0240          PLA          ;LEN(SS$)
0250          STA LENSS
0260          LDA #0
0270          TAY
0280          STA POSIT    ;INITIALIZE TO TOT$(0,0)
0290          STA POSIT+1
0300 NEXTCH   LDY #0       ;INCREMENT TOT$(X,X)
0310          INC POSIT
0320          BNE CHEKCH
0330          INC POSIT+1
0340 CHEKCH   LDA (ADTOT),Y ;COMPARE CHARACTER OF TOT$ WITH SS$
0350          CMP (ADSS),Y
0360          BNE NOMATCH ;GO TO NOMATCH IF NOT EQUAL
0370          INY          ;IF EQUAL INCREMENT Y TO CHECK NEXT CHARACTE
0380          CPY LENSS    ;IF Y=LEN(SS$) THE MATCH IS FOUND
0390          BEQ FOUND
0400          BEN CHEKCH   ;GO CHECK NEXT CHARACTER
0410 NOMATCH  LDA ADTOT+1 ;CHECK IF WE ARE AT THE END OF TOT$
0420          CMP ENDTOT+1
0430          BNE INC      ;IF NOT THEN GO INCREMENT TOT$
0440          LDA ADTOT
0450          CMP ENDTOT
0460          BEQ NOFIND   ;IF THE END OF TOT$ THEN MATCH NOT FOUND
0470 INC      INC ADTOT    ;INCREMENT TOT$
0480          BNE NEXTCH   ;START LOOKING FOR A MATCH AT NEXT TOT$(X,X)
0490          INC ADTOT+1
0500          BNE NEXTCH   ;START LOOKING FOR A MATCH AT NEXT TOT$(X,X)
0510 NOFIND   LDA #0
0520          STA POSIT    ;RETURN A ZERO TO BASIC
0530          STA POSIT+1
0540 FOUND    RTS          ;RETURN POS. OF SS$ IN TOT$ TO BASIC
```

```
0 REM BY C. MUELLER, HUNTSVILLE ACE
10 DIM TOT$(20000),55$(255)
20 TOT$="THIS IS A TEST OF A MACHINE L
ANGUAGE STRING SEARCH"
30 FOR I=1 TO 8
40 TOT$(LEN(TOT$)+1)=TOT$
50 NEXT I
60 TOT$(LEN(TOT$)+1)="ATARI COMPUTER"
70 ? "ENTER SEARCH STRING;:I.55$"
80 POS=USR(1536,ADR(TOT$),LEN(TOT$),AD
R(55$),LEN(55$))
85 IF POS=0 THEN ? "STRING NOT FOUND":
END
90 ? :? "'";55$;"'=TOT$(";POS;",";POS+
LEN(55$)-1;")"
95 END
100 FOR I=1 TO LEN(TOT$)
110 IF TOT$(I,I+LEN(55$)-1)=55$ THEN 1
50
120 NEXT I
130 STOP
150 ? I
```

Greg Menke: FUNCTION KEY (from Feb issue)

```
10 ;Function Key Assembly listing
20 ;
30 ;V2.0 12/12/83
40 ;By Greg Menke
50 ;
60 ;
70 ;Lising in Atari Assembler Editor
80 ;format.
90 ;
0100  *=$600
0110  LDA 12           ;get Lo byte of DOSINI
0120  STA DOSINI       ;and store it
0130  LDA 13           ;now get the Hi byte
0140  STA DOSINI+1     ;and store that too
0150  JSR INIT         ;initialize VBI and new RESET vector
0160  RTS              ;return to DOS
0170  BRK              ;end
0180 ;
0190 DOSINI .BYTE 0,0
0200 INDEX .BYTE 0
0210 ;
0220 RESET JSR INIT    ;fix the RESET vector
0230  JMP (DOSINI)     ;and return
0240 ;
0250 INIT LDX #ROUTINE/256        ;Hi byte of VBI
0260  LDY #ROUTINE&255            ;Lo byte of VBI
0270  LDA #6            ;use immediate mode
0280  JSR $E45C         ;and start it
0290  LDA #RESET&255    ;Lo byte of new RESET address
0300  STA 12            ;store it
0310  LDA #RESET/256    ;Hi byte of new RESET address
0320  STA 13            ;store that
```

```
0560 ;
0570 SELECT LDX #SELCT/256     ;Hi byte of SELECT message
0580  LDY #SELCT&255    ;Lo byte
0590  JMP PRINT         ;print it
0600 ;
0610 OPTION LDX #OPTON/256     ;Hi byte of OPTION message
0620  LDY #OPTON&255    ;Lo byte
0630 ;
0640 PRINT STX $CC      ;store Hi byte
0650  STY $CB           ;store Lo byte
0660  LDA #0 ;
0670  STA INDEX         ;clear the print index
0680 LOOP LDY INDEX     ;get the offset
0690  LDA ($CB),Y       ;now get the character
0700  INC INDEX         ;add 1 to offset
0710  CMP #0            ;is this the last one?
0720  BEQ DONE          ;yes, exit
0730  JSR $F6A4         ;no, print the character
0740  JMP LOOP          ;go back and get another character
0750 DONE JMP EXIT      ;exit
0760 ;
0770 STRT .BYTE 156,"You hit START",0
0780 SELCT .BYTE 156,"You hit SELECT",0
0790 OPTON .BYTE 156,"You hit OPTION",0
```

```
0330  RTS ;return
0340 ;
0350 EXIT JMP $E45F     ;exit VBI
0360 ;
0370 ROUTINE LDA 763    ;get ATASCII code for last character printed
0380  CMP #1            ;is it CTRL A?
0390  BNE EXIT          ;no, exit
0400 ;
0410  LDA 53279         ;get CONSOL value
0420  CMP #7            ;is anything pressed?
0430  BEQ EXIT          ;no, exit
0440 ;
0450  CMP #6            ;is START pressed?
0460  BEQ START         ;yes, goto START routine
0470  CMP #5            ;is SELECT pressed?
0480  BEQ SELECT        ;yes, goto SELECT routine
0490  CMP #3            ;is OPTION pressed?
0500  BEQ OPTION        ;yes, goto OPTION routine
0510  JMP EXIT          ;several keys are down, so exit
0520 ;
0530 START LDX #STRT/256       ;Hi byte of START message
0540  LDY #STRT&255     ;Lo byte of same
0550  JMP PRINT         ;go print it
```

Dice Probability (Con't)

```
128 IF R=51 THEN GOTO 30
130 IF R=52 THEN RUN
132 IF R=53 THEN END
134 ? :? :? :GOTO 100
150 ? "A":TRAP 184
152 ? :? :?
154 PRINT #2;"NUMBER","APPEARED","EXPE
CTED"
156 PRINT #2;2,T2,(INT((1/36)*NUMBER+0
.5))
158 PRINT #2;3,T3,(INT((2/36)*NUMBER+0
.5))
160 PRINT #2;4,T4,(INT((3/36)*NUMBER+0
.5))
162 PRINT #2;5,T5,(INT((4/36)*NUMBER+0
.5))
164 PRINT #2;6,T6,(INT((5/36)*NUMBER+0
.5))
166 PRINT #2;7,T7,(INT((6/36)*NUMBER+0
.5))
168 PRINT #2;8,T8,(INT((5/36)*NUMBER+0
.5))
170 PRINT #2;9,T9,(INT((4/36)*NUMBER+0
.5))
172 PRINT #2;10,T10,(INT((3/36)*NUMBER
+0.5))
174 PRINT #2;11,T11,(INT((2/36)*NUMBER
+0.5))
176 PRINT #2;12,T12,(INT((1/36)*NUMBER
+0.5))
178 PRINT #2;"  "
180 PRINT #2;"       Dice rolls = ";NU
M
182 PRINT #2;"       Total rolls= ";NU
MBER
184 CLOSE #2
190 GOTO 100
500 REM OPEN KEYBOARD
502 OPEN #1,4,0,"K:"
504 GET #1,R
506 CLOSE #1
508 RETURN
600 SETCOLOR 1,C,2:SETCOLOR 2,C,10:SET
COLOR 4,C,10
610 RETURN
1000 SAVE "D:DICEPROB"
10000 REM SAVE ROUTINE
```

THE GREEN FLASHER

(reprinted from the December, 1983 issue of Frederick ACE Newsletter)

If you read the last article, you now have (and I hope understand) a technique to change the screen colors in GR.0. I mentioned a "secret" reason for using one of the Atari's interrupt timers to initiate the routine. The secret, if you haven't guessed, is to include the option to cause the cursor to flash. Just think. A green screen with a flashing cursor — this is starting to sound like the option list on a 3270.

The new portion of the program will work using memory location 755 ($02F3 in hex). This location is mainly used to control the visibility of the cursor. POKE 755 with 1 and the cursor disappears. POKE a 0 back in and the cursor reappears. A side effect of the invisible cursor is the loss of the "inverse video" capability. As a side note — try POKEing 755 with 5. Not very useful, but interesting.

If we an change 755 back and forth between 1 and 0, we can cause the cursor to "flash". We already have a time driven interrupt in the screen color routine. We only need to add logic to flip the value of location 755 to flash the cursor. Keep in mind the side effect I mentioned. Any inverse video characters will flash along with the cursor. You may or may not appreciate this, but that's how it is.

I want to clear up some confusion from the "Green Screen" program. To get a good green on black you will have to lower the brightness level. At normal brightness the display is white on green.

This routine is relocatable. If you have some favorite area to stash machine language code, you merely need to change the address the BASIC loader uses to stash the DATA values. You must also change the address which gets loaded into the address vector for Timer2.

Listing 1 is the BASIC loader. Listing 2 is the Assembly language code equivalent. Please use standard care and save the program before you attempt to run it as any error in the DATA statements will likely cause the dreaded lockup. When you run the BASIC program, you will be asked for the color of your choice. Respond with the number of the color of your choice from the standard list of color values. You will next be asked to enter the number of flashes per second you desire. My preference is 5, but you can experiment and find your own favorite.

The Green Flasher will continue unaffected by GRAPHIC commands and DOS calls. It may be simply detached by pressing SYSTEM RESET. If detached, it may be restarted by X = USR(1536). Good luck and happy flashing.

— Steve Monn

Rats' Revenge

I haven't had time to put any special touches on this game, but it should provide a good foundation for any climbing-chase type games. The same approach could be used in a Pac-Man style game. In fact, that's the way this game started out but I think the climbing style is more interesting.

The screen is divided into a matrix of positions, (12X11) where changes of direction can take place. The array 'allowed' holds bytes describing which directions are permitted. The lowest four bits contain 'ones' for permitted directions which are compliments of the normal joystick directions. The high four bits indicate which type of girder/ladder combination is present at any intersection.

You can create your own scene and save it to disk. The 'play own' option will then recall it and you can play it at any level you select. I haven't figured out how to save under a user provided filename yet.

It is quite challenging to provide a proper chase algorithm. In the one I wrote, the rats seem to either ignore you or latch on for dear life. Once they caught on, they were impossible to shake and it was impossible to complete the screen if there was more than one dead end.

I compromised by skipping part of the algorithm at random times. The rats are made to increase their I.Q. with level by comparing a random number to one dependent on the present level. You may notice all three rat moving routines are identical and they should be able to be replaced by one using indexed variables. I tried, but each time I did, the program compiled but locked up when run. Rather than blame the language, I simply took this as an indication I'm still learning to use Action!

— Stan Ockers

KONG

This March issue of ACE contains the BASIC listing for the game of KONG. Kong has kidnapped your fiancee and he has placed her at the top of a partly constructed building and it is up to you to rescue her.

On your way up, you must jump over bricks and barrels while trying to avoid all the barrels Kong throws at you. Use Joystick 1 to make the man walk and climb the ladders. To jump, press the fire button and push the joystick in the direction you want to move. Jumping is not permitted while on a ladder or on a yellow oil slick.

Bonus points are scored by reaching the top of each level and points are scored for each ladder climbed. Each successive level becomes harder and scores higher points and larger bonuses.

Hint: The barrels do not go all the way to the end of a level before coming down, so keep an eye on them.

— Sidney Brown

ML SORT

(reprinted from the November, 1983 Huntsville Atari Users Group News Letter)

I want to show you how to use a machine language subroutine in BASIC. I've chosen a search routine which is short, easy to use, and extremely fast. Get your assembler/editor out and type in Listing 1. I use Synassembler. If you use the Atari Assembler/Editor, the only changes you need to make are the equates which use an " = " instead of "EQ." and Line 10, which should be changed to:

```
00010      * = $0600
```

Assemble the program, correct any syntax errors and save the source file. The save the object file from locations $0600 to $0660 according to your assembler manual's instructions.

Take a look at the source code listing. See all the PLA instructions at the beginning of the routine? They are taking the information off the stack as placed there by a USR call from BASIC. This means we must know exactly what parameters are to be passed from BASIC and their order. This routine is written to find a small string in a large string by comparing the Small search string to the same number of characters in the large string, starting with the first character in the large string and incrementing the start by one until a match is found or the end of the large string is reached. We want the BASIC program to pass the addresses and lengths of the large string and the search string to our ML routine. So we pull these from the stack (notice the order of most significant byte and least significant byte). We use the length of the large string (TOT$) to calculate the end of the string (ENTOT). Once we have these addresses and lengths the actual routine to do the search is very simple. The value returned to BASIC when the RTS is executed is the value at zero page addresses $D4 and $D5, which I am calling POSIT. Therefore we place the position of the search string within the large string into these locations and return to BASIC. If a match is not found we place a zero into $D4 and $D5.

Listing 2 is a test of the subroutine from BASIC. Reboot with BASIC, go to DOS and binary load the object file (cassette users refer to the method found in the Atari Assembler/Editor Errata Manual for loading object files). Go back to BASIC and type in Listing 2. Save the program before execution since a problem in the ML routine can cause a lockup. The BASIC program creates a long string of the same words over and over, and then puts "ATARI COMPUTER" at the end. Notice the parameters which must be passed in the USR call at Line 80. Run the program and search for "ATARI". It should take less than one second. Then type "GOTO 100" to see the same search performed with the BASIC search routing at Line 100. Be prepared to wait awhile.

This little routine can be used in almost any program requiring a search. You culd use it to search for anything in memory — not only within a string. Just pass the starting address for the search and the number of bytes to search in place of the ADR(TOT$) and LEN(TOT$). You can find multiple occurances of your search string by using the USR function over and over again — just changing the starting point for the search.

Of course, to use this search you must binary load the object file into memory first.

— C. Mueller

TIDBITS

Reading the Keyboard

This month I'll cover a few more PEEKs and POKEs and relate some methods of getting input from the computer in a program.

If you have an application where you want to have the cursor disappear, you can just POKE location 752 with a 1. To turn the cursor back on, simply POKE a 0 into 752.

In many programs I have written, I have found it necessary to read the keyboard in such a way as to produce immediate action upon pressing a key. If a simple INPUT statement is used, the computer will wait for the return key to be pressed before it continues. I know of two ways to avoid this delay. The first is a little more complicated than the second, and has both advantages and disadvantages. This method involves first OPENing an input/output control block for input from the keyboard. Then one simply uses a GET# statement to read the keyboard. This example program illustrates the technique:

```
10 OPEN #1,4,0,"K:":REM open the keyboard for input
 20 GET #1,A:REM the computer will put the ASC value of the key
pressed into the variable A. Note: It will not print anything on the
screen.
30 IF A = ASC("Y") THEN BRANCH ACCORDINGLY:REM compare the
keypress to what you desire.
40 GOTO 20:REM loop back for more
```

The advantage of this method is it gives us the ASC value of the keypress, so we don't need to do any translating. The disadvantage is the computer will stop and wait for a keypress, which may not be any good in some situations.

The other method requires the use of the internal code of keypress location (764). First you find out the internal code of the key you want the program to react to, then you simply test the value in location 764 to see if it is the one you want. To find the internal code, use this simple immediate-mode program:

```
FOR A = 0 TO 1 STEP 0:? PEEK(764):NEXT A
```

Now push the keys you want to use, and record the values the computer prints. Then in your program you just need to PEEK 764, compare it to the value you wanted, and branch accordingly. With this method the computer will never stop executing the program to wait for the keypress.

Reading the console keys (START, SELECT, and OPTION) is similar to the second method of reading the keyboard. In fact, one does the same thing exactly except the location used is 53279. Incidently, to clear location 764, POKE it with a 255, and to clear location 53279, POKE it with an 8. To make the little speaker inside the computer click, POKE 53279 with 7.

Next time I promise to deal with disk drive tricks.

— Dale Lutz
Canada

BYTES, BITS & NYBBLES

(reprinted from the August, 1983 issue of Keeping PACE)

I wish to present a tidbit of Machine Language (ML) and call your attention to a little known and poorly documented method of incorporating ML subroutines into a BASIC program. Let's start with a very simple ML routine and look at the various ways it may be incorporated into a BASIC program. the ML routine listed here does nothing more than add 5 to a value INPUT from the keyboard. It only works for single byte values (0-255) and wraps around so 255 + 5 = 4 (this is called MOD 256, but that's another story).

Machine Assembler Language
Key Strokes to Produce

Decimal	ML code	Hex code	mnemonic
104	PLA	68	(lower)[h]
24	CLC	18	[CTRL][X]
104	PLA	68	(lower)[h]
104	PLA	68	(lower)[h]
105	ADC	69	(lower)[i]
05	-05	05	[CTRL][E]
141	STA	9D	(Atari key) [CTRL][M]
00	$00	00	[CTRL][,]
06	$06	06	[CTRL][F]
96	RTS	60	[CTRL][.]

The general form of a USR function is: Z = USR([address of start of ML subroutine], [1st parameter], [2d parameter], [etc.]). When the USR call is made the OS loads the stack with the return address, passes the parameters (two bytes for each), and caps it off with a single count byte of the number of parameters passed. The above subroutine pulls off the count byte (PLA) from the stack, clears the carry bit of the status register (CLC) so we start with a clean slate, pulls off the most significant byte of the parameter passed from the stack (PLA) (we'll discard this value), pulls off the least significant byte of the parameter (PLA), adds (ADC) 5 (#05) to the lo byte of the parameter passed, stores the result (STA) into memory location 1536 ($00)($06), and returns from the subroutine.

To establish the ML routine in a BASIC program, you are generally instructed to do some variation of one of the following:

1. POKE it into a reserved memory area — i.e., page 6.

```
 10 GOSUB 100
 15 GRAPHICS 0:POSITION 5,5:? "INPUT A ";:INPUT A
 20 Z = USR(1537,A)
 30 POSITION 5,10:? A;" + 5 = ";PEEK(1536)
 40 END
 100 FOR X = 1 TO 10:READ BYTE:POKE 1536 + X,BYTE:NEXT
X:RETURN
 101 DATA 104,24,104,104,105,5,141,0,6,96
```

2. POKE it into a space above your BASIC program (making these changes to Program 1):

```
 5 DIM E$(1):REM E$ should be the last thing dimensioned
 20 Z = USR(ADR(E$) + 1,A)
 100 FOR X = 1 TO 10:READ BYTE:POKE ADR(E$) + X,BYTE:NEXT
X:RETURN
```

3. Assign it to a string by program control (making these changes to Program 1):

```
 5 DIM E$(10)
 20 Z = USR(ADR(E$),A)
 100 FOR X = 1 TO 10:READ BYTE:E$(X,X) = CHR$(BYTE):NEXT
X:RETURN
```

If you've used ML subroutines at all you are most likely familiar with these methods. Now try this:

```
 10 GRAPHICS 0:POSITION 5,5:? "INPUT A ";:INPUT A
 20 Z = USR(ADR(" ...*... "),A):REM "...*..." is the sequence of key
strokes as indicated in the table above
 30 POSITION 5,10:? A;" + 5";PEEK(1536): ?: ?
 40 END
```

You'll notice I PEEKed the answer. Normally "Z" will hold the answer (taken from 212,213 — lo,hi), but in this case "Z" is equal to the memory location of the first byte of the ML subroutine within the BASIC Statement Table. After all, if you've got the byte sequence set up in the BASIC Statement Table why write it out again in some other memory area? In addition, if you need to call the ML subroutine elsewhere in a program you can use the value of "Z" to locate it — USR(Z,A).

On pages 10-17 of "De Re Atari", number 19 states "Small assembly routines can be stored in USR calls" and gives one example. That's it — the total documentation of this elegant and memory-efficient method for calling short ML routines.

The advantages of this method are two fold: It saves time and memory by using the routine from the BASIC Statement Table (since you don't have to POKE the subroutine into memory); and the variable used to make the USR call (Z = USR(ADR("...")) can be used to make a later call (X = USR(Z)).

— John David McFarland III

VECTORS & ADDRESSES

(reprinted from the November, 1983 issue of the Atari Computer Club of OKC)

Now it's show & tell time for my latest how-to in ML. In spite of all the great things I said about the Synassembler last time, I am NOT selling my Assembler-Editor cartridge (Atari). I still have use for a VERY compact, simple Assembler in incorruptable ROM. This means it rarely crashes when I change an operating parameter.

As I've mentioned before, almost everything in the Atari Operating System (OS) is vectored. The reasoning is that if a ROM must be changed (revised) then every piece of software jumping to the locations may not be able to find the place it's supposed to go. Either the routine has been moved, or since it is not common to leave much extra space in ROM, it may have been moved around somewhat to make room for another routine which had to be added or fixed. The solution in the 400, 600, 800, 1200, 1400 series is to have pointers. The pointers (called vectors) are places holding the locations of the actual routines or hardware addresses. These are "guaranteed" not to be moved by Atari. So, even if it IS redundant to jump to a location merely to tell you to go to another location, there is a method to the madness. If something does get changed on the destination end (to which the vector points) the original software program doesn't need to be changed, just the address in the vector. All the original program has to do is remember where the vector is and go there, Atari will take care of the rest.

Another side benefit of this is there is really no reason a person who is aware of the operations can't change the vectors to point to their own routines first. For example, when a key is pressed, it generates an interrupt request to the CPU. This means "stop what you're doing and checkout the keyboard if you're not too busy." Normally, the CPU will then jump through or go to a vector telling it where the keyboard routine is in the OS. HOWEVER, put your own routine in, cange the pointer (or vector) to point to it, and instead you can go to your own routine.

WHY? As it infers in De Re Atari (the inspiration for this stuff): He who controls the keyboard controls the system. If the system goes to your routine 1st, your routine (another name for a short program) can change anything about keyboard input. My program simply checks to see if the key pressed is CTRL-W. If not, it goes back to where the original vector went. If it makes it through the 1st "filter" (IS a CTRL-W) then the rest of the routine executes.

Again, WHY? The reasoning behind this involves modems and me. At 300 baud I have no problems — if you want to stop everything from going by for a moment, you hit CTRL-S (called X-Off). This suspends the sending of data until you hit CTRL-Q (X-On) to go on. Fine! Now I work with a larger system using 1200 baud and 80 column lines. At 1200 baud the words are appearing on the screen (via the phone) 4x as fast and long lines of text don't end at the edge of the screen but wrap around, making some words hard to read. By the time I can look down at the keys and put my slow fingers on the CTRL and S keys, what I wanted to read — I said I'm slow! — is gone. So I redefined one of the keys as above so that if CTRL-W is pressed, it'll get through the filter and cause my code to check flags (to see what was sent last) and then tell the CPU (lie to it) and tell it the key pressed was either a CTRL-S or CTRL-Q (always opposite of the one sent last time). The next benefit of ths is I can now hit one key and the modem stops sending. Hit another and it resumes. Now, why go to all this trouble? Well, of course this is only a how-to. You don't have to do anything as insignificant as this. It's elegant and you don't have to modify or even know what the program does. Just pick any key it doesn't normally use and redefine it.

Only 3 other things ought to be mentioned. When the OS gets (and responds to) one of these interrupts it 1st pushes whatever value it was working on before interruption (the accumulator) onto a temporary storage location called "the stack". This is just a group of locations used for temporary storage. Before RTI (ReTurn from Interrupt) you should put it back as it was (PLA or PuLl Accumulator from stack). The Atari Assembler reserves 2 memory locations for an address when it sees a label, so what I did was LDA with 8 bits of the Vector destination and put it into part of the space left blank for JMP during the first pass of the Assembler. I then repeated this for the other half of the address (addresses are 16 bits long and the accumulator only 8). This is a very often used method of "moving a vector" because, even if Atari changes the ROMs and destination address, you're still going to wherever the vector said and you don't even have to know what the address in it was. The third note is that none of the codes for the keys are yet ASCII. The OS was in the process of getting info from the keyboard and that info is in the internal form or the actual value produced by the keyboard matrix. As always, there are lots of comments in the code to tell you play-by-play what's going on.

Parting Note: I don't charge for the programs I write in these pages (they're about as sellable as a board with a hole...HEY!...), but if you ever want to use a close derivative of one in your programs, newsletter, or magazine, I just love to see my name in print (I'm a ham), so you might mention 'lil ol me. Thank you.

```
10      ;VKEY.SRC—TOGGLES BETWEEN SENDING CTL-S OR
        CTL-Q (STOP & START FOR TELECOMM) WHEN CTL-W
20      ;(FOR CONTROL-WAIT) IS PRESSED. THIS WORKS BY
        RE-ROUTING KEYBOARD VECTOR
30      ;ADAPTED FROM PG. 8-15 OF DE RE ATARI
35      ;(NOTE THIS IS INTENDED AS AN AUTORUN FILE—NOT
        A BASIC USR CALLED FUNCTION
40      CH = $2FC
50      KBCODE = $D209
60      VKEYBD = $0208
65      ;!!!!NOTICE ORIGIN!!!!
70      * = $3FD ;I PUT IT IN CASSETTE BUFFER—IF YOU
        WANT TO USE CASSETTE—BETTER MOVE IT!
80      START SEI ;SO WON'T BE BOTHERED WHILE MOVING
        VECTOR (INSURANCE)
90      LDA VKEYBD ;STEAL KEY VECTOR
0100    STA JUMP + 1 ;& PUT IN AT END
0110    LDA VKEYBD + 1 ;OF MY CODE
0120    STA JUMP + 2
0130    LDA #REP&255 ;NOW GET ADR
0140    STA VKEYBD ;OF MY STUFF
0150    LDA #REP/256 ;& PUT IN INTERRUPT
0160    STA VKEYBD + 1 ;VECTOR FOR KEYBOARD
0170    CLI ;RE-ENABLE KEY INTERRUPTS
0180    RTS ;BACK FOR REST OF DOS IF AUTORUN FILE—EL
        CRASHO OTHERWISE (BUT ATARI DEBUG
        RECOVERS—SO OK
0181    ;TO TEST & USE WITH IT)
0190    ;*****************************
0210    REP LDA KBCODE ;CHK CODE
0220    CMP #$AE ;CTL-W?
0230    BNE JUMP ;NAH! SEND IT ON
0250    LDA FLGB ;SEE WHAT I SENT LAST TIME THROUGH
0260    EOR #$FF ;FLIP FLAG (DID IT 1ST SO I WOULDN'T
        HAVE TO DO IT FOR EACH CASE AFTER BRANCH)
0265    STA FLGB ;PUT NEW VALUE BACK! (00 OR $FF)
0270    BNE STP ;99• OF THE TIME, FLGB WOULD HAVE BEEN
        0 BEFORE XOR, SO I GO TO ROUTINE TO SEND CTL-S
0280    ;(DOESN'T MATTER A LOT WHAT FLGB WAS BEFORE
        THOUGH...ALWAYS SEND ONE OR THE OTHER —TOG-
        GLE— SO IF IT DOESN'T
0290    ;WORK 1ST TIME JUST PRESS CTL-W AGAIN)
0300    GO   LDA #$AF ;IF NOT STP-THEN
0310    STA CH ;SEND A CTL-Q
0315    PLA ;RESTORE A REG.(O.S. PUT IT ON STACK WHEN
        INTERRUPT OCCURED)
0320    RTI
0330    STP LDA #$BE ;SEND CTL-S
0334    STA CH
0338    PLA ;RESTORE A REGISTER (SEE ABOVE)
0340    RTI
0350    ;
0360    JUMP JMP JUMP ;REAL O.S. VECTOR GETS MOVED TO
        ARG. OF JMP HERE
0370    FLGB .BYTE 00 ;FLAG BYTE
0380    ;THIS IS LOAD 'N' GO VECTOR SO IT WILL START RUN-
        NING AS SOON AS ENTERED BY DOS
0400    * = $02E0
0410    .WORD START
```

Best of ACE books

Volume 1 are bound issues of the ACE Newsletter from the first issue, Oct 80 to June of 1982

Volume 2 covers July 1982 to June 1983

Only $12 each ($2 extra for Airmail). Available only from:

George Suetsugu
45-602 Apuapu St
Kanoehe, HI 96744

SortFinder 1.2
Composite index of Atari related articles from 5 popular computer periodicals from Apr '81 to June '83, including ACE. Only $6 for ACE member from:

Jim Carr, Valley Soft
2660 S.W. DeArmond
Corvallis, OR 97333

ATARI COMPUTER ENTHUSIASTS
3662 Vine Maple Dr. Eugene OR 97405

FIRST CLASS MAIL

Atari Computer Enthusiasts

A.C.E. is an independent, non-profit and tax exempt computer club and user's group with no connection to the Atari Company, a division of Warner Communication Company. We are a group interested in educating our members in the use of the Atari Computer and in giving the latest News, Reviews and Rumors.

All our articles, reviews and programs come from you, our members.

Our membership is world-wide; membership fees include the A.C.E. Newsletter. Dues are $12 a year for U.S., and $22 a year Overseas Airmail and include about 10 issues a year of the ACE Newsletter.
Subscription Dep't: 3662 Vine Maple Dr., Eugene, OR 97405.
**

President ..... Kirt Stockwell
4325 Sean , Eugene, OR 97402 / 503-689-5355
Vice Pres ..... Larry Gold
1927 McLean Blvd., Eugene, Or 97405 / 503-686-1490
Secretary ..... Bruce Ebling
1501 River Loop #1, Eugene, OR 97404 / 503-688-6872
Librarian ..... Ron and Aaron Ness
374 Blackfoot, Eugene, Or 97404 (503)689-7106.
Co-Editor ..... Mike Dunn
3662 Vine Maple Dr., Eugene, Or 97405 / 503-344-6193
Co-Editor ..... Jim Bumpas
4405 Dillard Rd., Eugene, Or 97405 / 503-484-9925
E.R.A.C.E. (Education SIG Editor) ..... Ali Erickson
295 Foxtail Dr., Eugene, Or 97405 / 503-687-1133
E.R.A.C.E. Corresponding Secretary ..... Robert Browning
90 W. Myoak, Eugene, OR 97404 / (503)689-1513

Send 27c stamps or coin (50c overseas) to the Ness' for the new, updated ACE Library List —new in Feb 84!

Bulletin Board
(503) 343-4352

On line 24 hours a day, except for servicing and updating. Consists of a Tara equipped 48K Atari 400 with a TARA keyboard, 2 double-density double sided disk drives with an ATR 8000 interface, 2 double density Percom disk drives, an Epson MX80 printer, a Hayes SmartModem; running the ARMUDIC Bulletin Board software written by Frank L. Huband, 1206 N. Stafford St., Arlington, VA 22201. See the Nov '82 issue for complete details.